Thus the circuit has a reduced feedback factor without resorting to increasing its gain.

For the components in the **figure**, the overall feedback factor β is 0.09, which has the same stabilizing effect as raising the circuit's gain to 11. Overshoot is now a residual 0.2% resulting from feedback-delay oscillations.

Adding R_1 and R_2 does not alter the circuit's gain because the input signal does not appear across these resistors. For unity gain, the voltage across R_2 must, after all, be only the error signal. This error voltage injects a small current into R_1. The resulting voltage differences on R_1 and R_2 define an input-to-output voltage difference equal to $(V_{OS}+V_{OUT}/A)/\beta$. This difference is small because $V_{OUT}\approx V_{IN}$ during continued operation.

This analysis also demonstrates two other effects of the positive feedback. First, the circuit amplifies the input-error signal of IC_1 by $1/\beta=11$ rather than by unity. Second, the circuit's input resistance is much higher than you would expect. An input signal driving a feedback network normally suggests a low input resistance for an op-amp circuit. However, the positive feedback bootstraps the circuit's resistance in the critical sample interval. The current R_1 draws from the input equals $V_{OUT}/A\cdot R_2\approx V_{IN}/A\cdot R_2$. Thus, the circuit's input resistance is $A\cdot R_2$ during the sample interval. **(EDN BBS /DI_SIG #944)** EDN

To Vote For This Design, Circle No. 748

8051 program converts BCD to binary

John T Hannon
Stewart Warner Alemite, Charlotte, NC

The assembly language program in **Listing 1** for 8051-family single-chip μPs converts a BCD number to binary. Unlike other special-purpose, BCD-to-binary routines, you can easily extend this program to convert numbers larger than the 5-byte BCD numbers allowed here.

Before kicking off the program, you must store the 5-byte BCD number in a 5-byte register and allocate a 2-byte register for the binary result. During initialization, the program sets up pointers to these two registers and clears the binary register. Then the program stores the first BCD digit in the result register.

The program has two routines. After initialization, the first routine sets up the conversion factor for the first bit of each BCD digit. The initial value for the

Listing 1—8051 BCD-to-binary conversion program

```
LABEL    MNEMONIC                    COMMENT

BCDBIN:  MOV      R0,#REGBCD         ;BCD REGISTER POINTER
         MOV      R1,#REGBIN         ;BINARY REGISTER POINTER
         MOV      @R1,#00H
         INC      R1
         MOV      @R1,#00H           ;ZERO BINARY REGISTER
         DEC      R1
         MOV      A,@R0
         ANL      A,#0FH             ;SAVE FIRST BCD DIGIT
         MOV      @R1,A
         MOV      R4,#0AH            ;CONVERSION NUMBER FOR FIRST BIT
         MOV      R5,#00H            ;OF SECOND BCD DIGIT (10)
         CALL     BCDCON
         MOV      R4,#64H            ;CONVERSION NUMBER FOR FIRST BIT
         MOV      R5,#00H            ;OF THIRD BCD DIGIT (100)
         CALL     BCDCON
         MOV      R4,#0E8H           ;CONVERSION NUMBER FOR FIRST BIT
```

Listing 1—8051 BCD-to-binary conversion program *(continued)*

```
        MOV     R5,#03H         ;OF FOURTH BCD DIGIT (1000)
        CALL    BCDCON
        MOV     R4,#10H         ;CONVERSION NUMBER FOR FIRST BIT
        MOV     R5,#27H         ;OF FIFTH BCD DIGIT (10000)
        MOV     R2,#03H         ;CHECK ONLY 3 BITS OF FIFTH DIGIT
        CALL    BCDCN1
        RET

BCDCON: MOV     R2,#04H         ;CONVERSIONS PER BCD DIGIT
BCDCN1: INC     R0              ;INCREMENT BCD REGISTER TO NEXT DIGIT
        MOV     A,@R0           ;GET BCD DIGIT
        ANL     A,#0FH          ;MASK OFF UPPER FOUR BITS
        MOV     R3,A            ;AND STORE IN TEMPORARY REGISTER
BCDCN2: MOV     A,R3            ;GET BCD DIGIT
        RRC     A               ;SHIFT DIGIT ONE BIT RIGHT
        MOV     R3,A            ;STORE SHIFTED DIGIT AGAIN
        JNC     BCDCN3          ;IF NO CARRY, DO NOT ADD FACTOR
        MOV     A,@R1           ;GET FIRST BYTE OF BINARY RESULT
        ADD     A,R4            ;ADD FIRST BYTE OF CONVERSION FACTOR
        MOV     @R1,A           ;STORE FIRST BYTE OF BINARY RESULT
        INC     R1              ;INCREMENT TO 2ND BYTE OF BINARY REG
        MOV     A,@R1           ;GET SECOND BYTE OF BINARY RESULT
        ADDC    A,R5            ;ADD (WITH CARRY) 2ND BYTE OF FACTOR
        MOV     @R1,A           ;STORE SECOND BYTE OF BINARY RESULT
        DEC     R1              ;DECREMENT POINTER TO FIRST BYTE OF
                                ;BINARY REGISTER

BCDCN3: MOV     A,R4            ;DOUBLE VALUE OF CONVERSION NUMBER
        CLR     C               ;FOR EACH BCD DIGIT BY SHIFTING R4
        RLC     A               ;AND R5 ONE BIT TO THE LEFT.
        MOV     R4,A
        MOV     A,R5
        RLC     A
BCD12:  MOV     R5,A
        DJNZ    R2,BCDCN2       ;DECREMENT CONVERSION COUNTER
        RET                     ;REPEAT UNTIL ALL BITS CHECKED
```

conversion factor is 10_{10} ($000A_{HEX}$). Then the first routine calls the second routine as a subroutine to do the actual conversion. If the BCD digit's bit is a one, the subroutine adds the conversion factor for that bit to the partial binary result. Then the program adjusts the conversion factor to correspond to the next bit in the selected BCD digit. The program checks each bit in a similar fashion.

The program uses a loop counter to check all four bits of each BCD digit. When the loop counter reaches zero, the program increments the BCD-register pointer, gets the next BCD digit, and masks off the digit's upper four bits in case the number is in ASCII. The program then shifts one bit to the right through the carry bit and stores the shifted number.

Checking the carry bit determines if the conversion factor gets added to the binary result or not. If the bit is a zero, the conversion factor does not get added; if the bit is a one, the conversion factor does get added.

To repeat the bit-conversion loop, the program shifts the conversion factor one bit to the left, doubling its value. The result of decrementing the bit counter determines if the loop needs to be repeated or not. This test allows the conversion routine to test each bit in a BCD digit and add 10_{10}, 20_{10}, 40_{10}, and 80_{10} for each bit, respectively, that is a one.

After converting the first digit, the program returns to the initialization section, adding 100_{10} (00064_{HEX}) to the conversion factor. The program then repeats, converting the second, third, and fourth BCD digits. For the fifth—or ten-thousands—digit, the bit counter's value is only 3 instead of 4 because a 5-digit BCD number cannot be greater than 65,535.

You can easily extend this program by expanding the registers and extending the counters. You can obtain the **listing** from the EDN BBS (617) 558-4241, 300/1200/2400, 8, N, 1—from main menu, enter (s)ig, <s/di_sig>, rk941). **(EDN BBS /DI_SIG #941)**

EDN

To Vote For This Design, Circle No. 749